UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Generació de props i entorns procedurals per a videojocs.

Treball Final de Grau

Grau en Disseny i Desenvolupament de Videojocs

Cognoms: Olóndriz Rosique Nom: Daniel

Pla: 2014

Director: Borrás Borrell, Elías

Índex

Resum

El procés de desenvolupar grans entorns en videojocs es un procés lent i tediós. Actualment el procés més estàndard de la industria és anar posant en un entorn diferents peces o objectes modulars (que encaixen entre elles) per tal de construir un escenari variat. El que es proposa amb aquest treball de fi de grau és desenvolupar una eina que permeti al dissenyador o artista d'entorns anar més ràpid a l'hora de desenvolupar la seva feina, sense que aquest hagi de canviar completament el seu mètode de treball. Això s'aconseguirà creant una eina que a partir dels inputs que li proporcioni l'usuari, pugui crear proceduralment aquests assets, centrant-se en el desenvolupament d'edificis.

Per tal de dissenyar i desenvolupar l'eina s'ha utilitzat el software de Houdini, i s'ha fet una recerca extensa sobre quins paràmetres ha de necessitar un dissenyador o artista per crear edificis realistes.

Com a metodologia s'ha fet servir la metodologia àgil de scrum, i s'ha seguit la planificació proposada en un diagrama de Gantt.

Paraules Clau

Tool, Procedural, Generation, Videogame, Development, Environment, Building, Props

Enllaços

- Enllaç al tràiler resum del treball: https://www.youtube.com/watch?v=u-TQjQpQ0mI
- Enllaç al arxiu otl de la eina: https://mega.nz/#!t7hn1YaY!jG4OexnfGu4LoFxOtdla606BW7EvsbIDZvAc1Nfrqac

NOTA: Per que la eina funcioni es necessita tenir Houdini, Houdini Engine i Unreal Engine instal·lats. A part es necessita una llicència de Houdini Indie o superior.

Índex de Taules

Índex de Figures

Glossari

- **Generació Procedural/Procedimental:** és un mètode de creació de contingut basat en algoritmes matemàtics, en comptes de ser creat per un usuari de forma manual.
- **Environment:** Paraula que es fa servir en el sector dels videojocs per referir-se als entorns i nivells dels jocs.
- **Prop:** Objectes que es troben en el entorn de un videojoc i que poden ser usats per el jugador.
- **Asset:** Paraula que es fa servir en el sector dels videojocs per a definir el contingut que hi ha dins de un videojoc, poden ser des de models 3d, textures o sons. Bàsicament és qualsevol contingut creat per els artistes o dissenyadors del projecte.
- **Texturitzat:** El procés de crear textures per a un model 3d.
- **Workflow:** sinònim per mètode o seqüència de treball. En el sector dels videojocs, els passos que ha de seguir un asset per arribar a estar dins del joc (modelat, texturitzat, etc.) . En el sector dels videojocs s'acostumen a tenir workflows pre-definits per a la creació de contingut.
- **Textures de soroll o "noise":** textures que han sigut generades a partir de un algoritme i que per tant son procedurals. En el entorn multimèdia s'usen com a base per crear més contingut procedural.
- **Software:** Sinònim de programa informàtic.
- **Dungeon:** Masmorra.
- **Unreal Engine:** Nom de un motor de videojocs bastant popular
- **Blueprints:** Sistema de programació visual pròpia de Unreal Engine.
- **Unity:** Nom de un altre motor de videojocs.
- **Tile:** Traducció literal de rajola. En el sector dels videojocs per parlar assets amb una forma particular que es poden connectar amb altres com si fossin peces.
- **Tileset:** Conjunt de assets que estan fets per poder-se connectar entre si per tal de fer assets més grans. Exemple: conjunt de parets, portes i finestres que et permet crear diferents cases.
- **Tool:** Eina o programa senzill que te una sola finalitat.
- **Seed:** Número a partir del qual es generen paràmetres aleatoris.
- **Random:** Aleatori.

- **Plugin:** petit programa que afegeix funcionalitats addicionals a un altre programa.
- **PlaceHolder:** Recurs o objecte que s'utilitza sol per a testejar.
- **Instanciar:** Posar una copia de un objecte
- **Houdini:** Programa especialitzat en generació de partícules i models procedurals.
- **Houdini Engine:** Plugin que permet exportar assets de Houdini a Unreal Engine o Unity.
- **Modular:** Un element modular és un element que s'usa repetides vegades, i que junt amb altres elements modulars, es poden construir assets més grans.

1. Introducció

1.1. Motivació

La generació de contingut procedural sempre ha sigut una cosa que m'ha interessat dins del sector dels videojocs. Venint de un perfil de artista tècnic, crec que és molt interessant veure com els artistes i desenvolupadors se les havien enginyat per fer coses que seguint un procés tradicional no s'haguessin pogut aconseguir. Jocs com *Dead Cells*(Motion Twin, 2017), el qual cada cop que el jugues es diferent, o *Spore*(Maxis, 2008), on els mons son generats proceduralment m'han fet tenir un cert interès a veure com es van desenvolupar i quines tècniques van fer servir.

Com he comentat m'interessa molt el perfil de artista tècnic ja que es un perfil que justa la part artística que m'apassiona dins de un videojoc, i la junta amb la part tecnològica que es la que fa possible la anterior. Veient que la generació de contingut procedural (la qual és una part molt tècnica) es podia utilitzar en el àmbit artístic, em va interessar fer una ullada per veure de que es tractava.

1.2. Formulació del problema

La majoria de jocs actualment estan formats per un nombre elevat nivells i escenaris, o per grans entorns (mons oberts, etc.).
Dins d'aquests nivells o entorns es reutilitzen molts assets per diferents parts d'aquests ja que el cost i sobretot el temps i el personal necessari per generar una quantitat d'assets d'aquesta magnitud seria massa elevat. Com a estàndard doncs, el procés que es segueix és que els Environment o Prop Artists creen un pack/número determinat de assets i props, i aquests es reutilitzen per els diferents nivells amb la mateixa estètica dins del joc, o en les diferents localitzacions. Això causa la repetició de assets com cases, entre altres.

Figura 1. Casa i les seves localitzacions en el mapa de Fortnite

Actualment gràcies a la aparició del modelatge procedural, s'ha pogut substituir el procés estàndard per un nou procés on els assets que es creen s'adapten al entorn gràcies a la seva creació procedural. El principal inconvenient d'aquest procés és que la solució que presenta acostuma a ser útil sol per al projecte que s'ha creat, ja que s'ha desenvolupat amb aquesta finalitat, aleshores el resultat no és extensible a altes projectes. A part d'això, es requereix que el desenvolupador aprengui un nou workflow completament diferent al que tenia establert.

El problema que es proposa solucionar amb aquest tfg, doncs, és el de la manca de una eina més generalista que serveixi per una gran varietat de videojocs que permeti reduir el temps de desenvolupament i la repetibilitat del entorn.

1.3. Objectius generals del TFG

- **Desenvolupar una eina que agilitzi el procés de creació d'entorns:** Seguint l'apartat anterior doncs, l'objectiu d'aquest treball és desenvolupar una eina la qual proporcionats una llista de assets, creï diferents assets d'una manera procedural, agilitzant el procés del Level Designer o Environment Artist, optimitzant el temps de creació d'entorns a la vegada que reduint la repetibilitat de assets.

  Posem per exemple que s'ha de crear una ciutat. Mentre que el procés o workflow tradicional seria crear els diferents edificis un per un a partir de assets com portes, parets i finestres, doncs amb la eina que es planteja, passant-li sol els assets, sigui capaç de **generar-te les diferents cases a partir de algoritmes.** La principal fortalesa de la eina és que es pugui **adaptar a molts escenaris diversos**, des de un poble medieval fins a una ciutat cyberpunk.

1.4. Objectius específics del TFG

- **Dissenyar una eina útil per a dissenyadors i artistes:** Aconseguir dissenyar una eina que permeti a artistes i dissenyadors usar-la, sense tenir coneixement previ de programació, per crear contingut procedural.
- **Entrar en detall en la creació de contingut procedural:** Analitzar i entendre com funcionen els algoritmes per generar contingut procedural i intentar aplicar-los dins d'aquesta eina.

1.5. Abast del projecte.

La eina que es planteja desenvolupar va enfocada als professionals del sector que no tinguin un perfil tècnic i que vulguin desenvolupar entorns procedurals o que vulguin crear entorns en general. Utilitzant aquesta eina, no fa falta aprendre un nou workflow o mètode des de 0, ja que és adaptable al ja existent, amb la qual cosa molts professionals que no sàpiguen el mètode de modelatge procedural també el podran usar. Com el exemple que s'ha comentat en el apartat 1.3, el ús de la eina estarà enfocat a reduir el temps de producció a la hora de fer un escenari de un videojoc, la qual cosa també permetrà pujar la qualitat dels jocs, ja que amb el mateix temps de desenvolupament es tindrà temps a fer mes quantitat.

La eina es centrarà doncs en el **desenvolupament de edificis de forma procedural**. A partir de uns tilesets ja creats, la eina podrà crear qualsevol tipus de edifici, i el seu objectiu o punt fort és ser el **més amplia possible**. És a dir que serveixi tan com per un projecte amb una ambientació de una ciutat metropolitana com Nova York, o per fer cases medievals com les de Skyrim.

Un futur projecte podria ser que la eina pugues desenvolupar escenaris sencers a partir del input dels dissenyadors i artistes, però tenint en compte el temps i la complexitat del projecte i la falta de pràctica en el tema, s'ha enfocat la eina al desenvolupament de assets per entorns.

2. Estat de l'art

2.1. Generació Procedural de Contingut

La generació procedural de contingut també conegut com PCG (Procedural Content Generation) és un mètode on el contingut es crea a partir de algoritmes matemàtics en comptes de manualment. Aquest mètode es pot trobar en forma de malles 3d, sons i textures entre d'altres i té moltes i diverses aplicacions.

En el sector dels videojocs, tot i que la utilització de eines procedural s'ha popularitzat en els últims anys i s'ha expandit el seu us per a modelar o texturitzar, és una tècnica que s'ha fet servir amb diferents aplicacions des de fa més de 2 dècades. El primer joc en fer servir generació procedural va ser "Elite" l'any 1984, el qual podia generar 8 galàxies amb 256 sistemes solars cada una.

En els últims anys trobem molts més exemples a la indústria que fan servir elements procedurals per als seus jocs en major o menor mesura i amb aplicacions bastant variades. El exemple més important pot ser "No Man's Sky" (2016), el qual és un joc el qual genera un univers de manera procedural on totes les criatures i planetes son diferents. Tot l'univers generat per aquest joc esta creat per algoritmes i els dissenyadors intervenen molt poc en la creació de aquest. Imatge treta del Article de Kotaku (Alexandra, 2016)

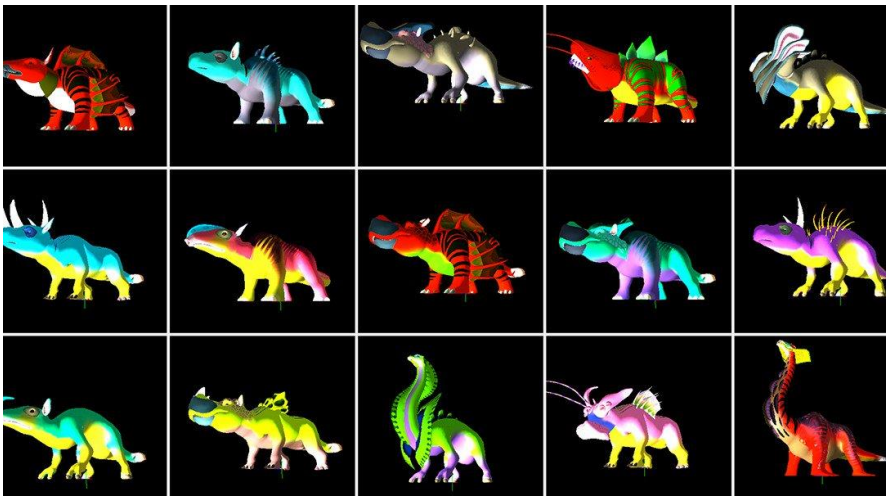


Figura 2. Animals generats proceduralment de No Man's Sky

Un altre exemple completament diferent el trobem amb “Far Cry 5” (2018) i “Far Cry New Dawn”(2019). Aquests dos jocs tot i tenir un món obert controlat i dissenyat per els dissenyadors, fan servir eines procedurals per la generació de aquest món. Tots els boscos i camps estan generats d’una manera procedural, i utilitzen el modelatge procedural per a assets com ponts, tanques o carreteres. Aquests dos últims son uns grans exemples de cos ha implementat el modelat procedural dins del sector dels videojocs. (Carrier, 2018)

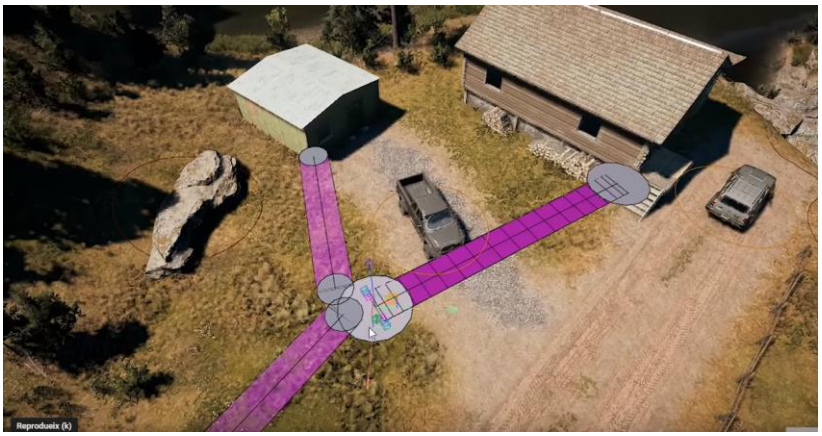


Figura 3. Eines de modelat procedural del Far Cry 5

2.2. Houdini Software

Una de les causes per les quals no apareguessin jocs que fessin servir modelatge procedural abans era la manca de software que permetés al usuari fer contingut modular i procedural. Houdini, un software que en un principi estava centrat en generar efectes de partícules per a pel·lícules, ha omplert aquest buit gràcies a les seves ultimes versions les quals permeten al usuari generar contingut procedural extraïble a motors de videojocs gràcies a la extensió de Houdini Engine. Aquest nou workflow de modelat ha permès el desenvolupament de jocs com els anteriors mencionats “Far Cry 5” i “Far Cry: New Dawn” o “Ghost Recon WildLands”. El principal avantatja és que els assets que genera son modulars i adaptables per moltes situacions. El principal desavantatge és que el workflow que es planteja Houdini substitueix al ja existent, i no el complementa, i per tant la majoria de tools desenvolupades en Houdini requereixen un nou aprenentatge.

Tot i així es podrien plantejar eines desenvolupades en Houdini que complementessin al workflow clàssic, però de moment no hi ha cap de especial rellevància. Es per això que una de les opcions a la hora de desenvolupar aquest treball de fi de grau sigui usar Houdini per desenvolupar la eina o tool.

Una altra desavantatge que te Houdini és el fet de que el seu aprenentatge requereix un perfil molt tècnic. És per això que no s'ha estandarditzat encara, ja que per usar-lo és necessari perfils artístics tècnics, i mentre que un artista pot tenir facilitat en aprendre noves eines i software de modelat tradicional, li serà molt més difícil aprendre aquesta aproximació, ja que és necessari un coneixement previ de programació i codi. Es per això que moltes vegades Houdini es usat més per programadors que per artistes.

Figura 4. Modelat procedural d'una casa en Houdini (Opara, 2016)

2.3. Substance Designer

Substance Designer és el equivalent a Houdini, però en el apartat de generació de textures procedurals. Es tracta de un programa el qual funciona a partir de nodes, i permet al usuari crear textures a partir de gradients, formes simples o textures de soroll generades proceduralment (Perlin Noise, Voronoi Noise,etc.). Desde la seva aparició s'ha convertit en un estàndard en la industria per a la creació de entorns.

2.4. Estudi de Mercat

La eina que es planteja desenvolupar es una eina amb la qual passats uns assets ja existents (parets, portes, sostres, props...) et pugui generar diferents entorns basats en algoritmes. El objectiu de aquest producte és ser tan generalista com sigui possible per tal de que sigui útil per una gran quantitat de jocs. Com a principals competidors a aquesta idea podem trobar:

2.4.1. Extensions per motors de videojocs:

- **Dungeon Architect:** Una extensió per a Unity i Unreal Engine 4, el qual serveix per crear “dungeons” o masmorres de una manera procedural basat en un sistema de tiles. És una idea bastant similar a la que es planteja en aquest treball de fi de grau, però mentre aquest projecte es centra en fer edificis per qualsevol tipus d’entorns, Dungeon Architect es centra en generar masmorres procedurals a partir de tiles predefinides. El principal avantatge d’aquesta eina és la profunditat que té a la hora de generar contingut procedural, ja que té moltes opcions i una interfície molt intuïtiva per desenvolupar. Per contra, el principal desavantatge es que sol ens permet crear dungeons/ masmorres, amb la qual cosa, si el teu joc no es basa en estructures de masmorres, aquesta eina no és útil.
- **BuildR2:** Extensió per a Unity que permet crear edificis de forma procedural. Mentre que el concepte és el mateix que el que plantegem, l’aproximació es bastant diferent. La eina que ens plantegem ha de ser una eina que complementi el workflow tradicional, i per tant li donem la llibertat al usuari que faci servir les seves pròpies tiles.

  BuildR2 en canvi té uns estils predefinits dels quals pots escollir i **no funciona a base de tiles**, la qual cosa és un gran inconvenient. Encara que té una molt bona eina de edició de edificis, deixa poca llibertat al usuari a la hora de escollir quina geometria posar a les parets, limitant-te majoritàriament a les que et presenta la eina, les quals no tenen un acabat molt professional.
- **Procedural Toolkit:** Extensió de Unity que permet generar terrenys i props i personatges bàsics de manera procedural. És una eina molt global que per el que es mostra, sembla possible generar molt contingut molt diferent. La principal desavantatge d’aquesta eina es que falta documentació o vídeos que expliquin mínimament com va o de que es tracta, ja que no s’explica si sol es tracta de algorismes procedurals, o quina aproximació té respecte a les motles coses que engloba.

2.4.2. Assets generats amb Houdini Engine

Gràcies a la naturalesa dels assets creats amb Houdini Engine, aquests ja son completament modulars i s'adapten a l'entorn. El principal inconvenient de la majoria d'aquests assets és que s'han creat des de zero amb Houdini amb la mentalitat de fer servir un workflow diferent al clàssic, amb la qual cosa, la majoria d'aquests no són escalables a projectes diferents, ja que si, per exemple, s'ha fet una casa medieval procedural, segurament s'ha fet amb la fi de crear un joc medieval, i al basar-se en un model creat des de 0 a Houdini i no en tiles, no és possible canviar aquesta casa a un edifici de Nova York, amb la qual cosa, aquest asset no ens serviria per altres jocs. Per contra, el principal benefici d'aquests assets és que poden ser usables per a varis motors de videojocs, i al ser naturalment procedurals, es poden reutilitzar tantes vegades com es vulgui dins de un projecte.

2.4.3. Engines

2.4.3.1. Unreal Engine

Per internet es poden trobar tutorials de blueprints fets amb Unreal que permeten fer coses similars, però la principal desavantatge d'aquesta opció es que s'ha de saber programar mínimament per desenvolupar la eina. En el apartat **5.2 Software a utilitzar**, s'explica amb més detall les avantatges i desavantatges que tindria desenvolupar una eina com al que es vol fer en aquest motor.

2.4.3.2. Unity

Unity permet també crear scripts de la mateixa forma que Unreal, però al igual que aquest, per generar tal script és necessari saber programar. Es poden trobar varis tutorials a internet o llibres com "Procedural Content Generation for Unity" de Ryan Watkins. En el apartat **5.2 Software a utilitzar**, s'explica amb més detall les avantatges i desavantatges que tindria desenvolupar una eina com al que es vol fer en aquest motor.

3. Gestió del projecte

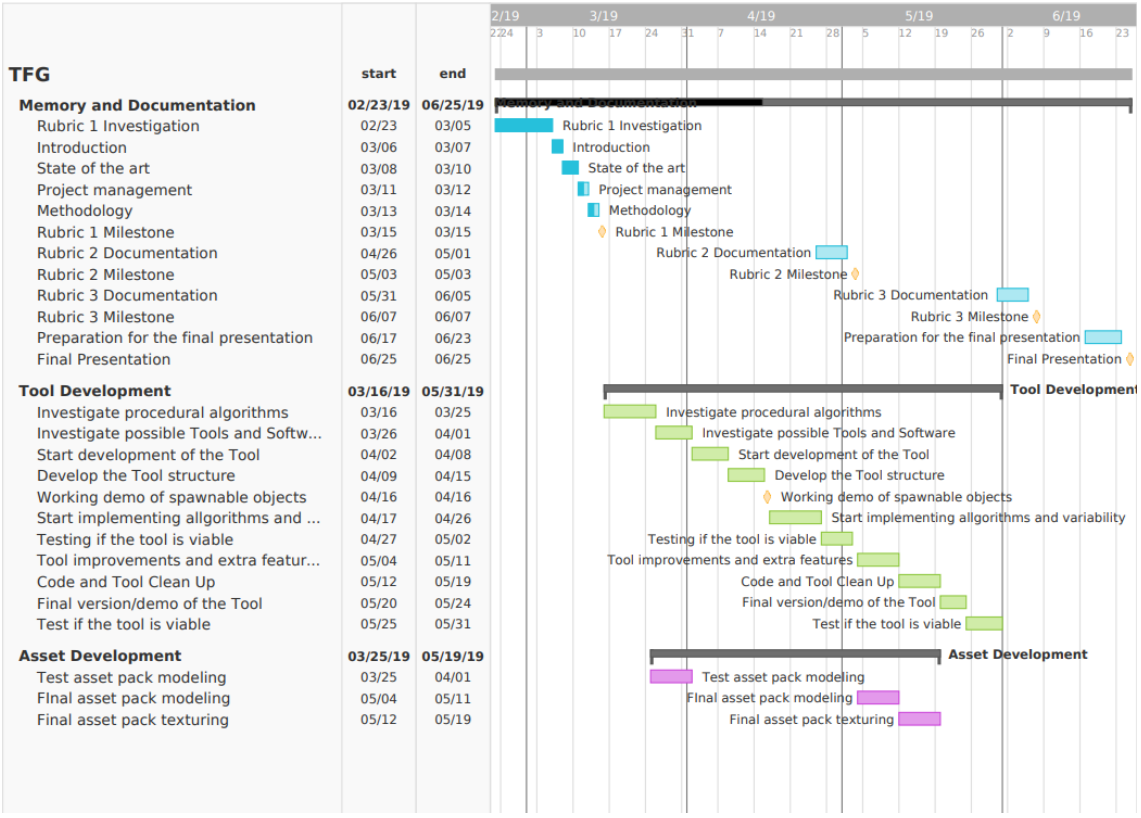
3.1. Procediment i Eines per al seguiment del projecte

En el sector del videojoc s'acostuma a dividir el temps de producció en 3 fases. Preproducció , Producció i Post-Producció. Tot i així aquest treball es centra en crear una eina, i no és completament necessari seguir aquest sistema. En comptes d'això, s'utilitzarà un sistema àgil basat en esprints. Alguns d'aquests esprints formaran part de investigació o preproducció ,mentre que la majoria estaran en el apartat de producció i finalment també hi haurà alguns de millores i poliment que els podrem englobar en una postproducció.

3.1.1. Diagrama de Gantt

Per planificar i dividir totes les tasques s'ha utilitzat un diagrama Gantt el qual ens mostra el temps que tenim plantejats per cada tasca. S'ha dividit en 3 grans grups:

- **Memory and Documentation**: El temps i tasques definides per a fer la documentació relacionada amb la memòria i la presentació final.
- **Tool Development**: El temps i les tasques que van dirigides a la creació de la eina en si.
- **Asset Development**: El temps i les tasques que van dirigides a la creació de assets que siguin vàlids per a aquesta tool.



Taula 1. Diagrama de Gantt del TFG

3.2. Eines de validació

Per validar el meu progrés en aquest treball, es realitzaran dues validacions.

La primera es farà per la entrega de la Rúbrica 2, i es tractarà de un procés intern de validació que faré jo mateix com a usuari, per tal de veure si estic complint amb els esprints marcats, i per veure si de moment la eina que s'està desenvolupant és mínimament funcional i compleix les funcions bàsiques.

La segona validació serà la més important i és la que es farà un cop a eina ja estigui acabada, i es tractarà de una prova. Primer de tot es mostraran un cert nombre de edificis recreats amb la eina a dins de un motor de videojocs i se li demanarà a companys de la industria amb perfil artístic o de dissenyador que intentin recrear aquests varis edificis, primer amb la eina (amb una explicació prèvia de com funciona) i després amb el mètode clàssic. Un cop realitzades les dues proves es compararà el temps que han tardat per veure si han anat més ràpid, i se'ls hi preguntarà si han trobat alguna cosa poc intuïtiva de eina per tal de guardar-ho com a feedback.

A partir d'aquí podrem comprovar si la eina compleix els objectius proposats, i podrem també veure el marge de millora en certs aspectes de la pròpia eina.

3.3. DAFO

Debilitats:

- **Falta de coneixement previ:** Tot i que s'ha fet una recerca prèvia per el treball, la manca de experiència i coneixement en aquest sector pot ser un factor decisiu que pot causar endarreriments i errors no previstos. És per això que s'han creat ja plans de contingència per tal de evitar retards imprevistos.
- **Massa generalista:** Si la eina es fa massa generalista, pot passar que no sigui útil per la majoria de jocs. És per això que a la hora de desenvolupar-la es tindrà en compte el veure si la eina és útil en diferents entorns i situacions.
- **Poc intuïtiu:** Depenent de quina eina es faci servir, es possible que faci la interfície poc intuïtiva i que la eina perdi part de la utilitat. Per tal de evitar que no s'entengui, quan es facin els testeigs de la eina final també es mirarà amb quines dificultats es troba l'usuari.
- **Temps de desenvolupament curt:** poc més de 3 mesos per desenvolupar una eina com aquesta pot resultar un període de temps curt. És per això que es plantejarà una eina simple o base que tingui possibilitats de expansió, si en un futur es vol desenvolupar amb més profunditat.

Fortaleses:

- **La optimització es un bé preuat:** Una eina que ajudi a optimitzar temps i per tant a reduir costos, es una eina que es valora molt dins del sector dels videojocs, ja que el temps sempre es un pilar dels projectes. Per tal de assegurar-nos de que la eina és útil, es demanarà a companys que utilitzin la eina per fer varis edificis, i que també els facin amb el workflow tradicional. D'aquesta manera es podrà comprovar el temps optimitzat.
- **Manca de eines generalistes:** Tot i que hi ha eines similars, no hi ha cap que et permeti generar edificis amb un ventall tan gran. La eina que es vol plantejar vol ser una eina útil que puguin fer servir varis estudis a la hora de realitzar el seu joc, i que no sigui una eina que sol es desenvolupa amb la fi de ferla servir en un sol joc.

Amenaces:

- **Eines i extensions similars:** Hi ha eines que, tot i ser més específiques, tenen resultats similars a la que es planteja. Aquestes eines es poden trobar a dins del punt **2.4 (Estudi de Mercat).**

Oportunitats:

- **Baix risc:** al tractar-se de un treball de fi de grau, no hi ha grans despeses econòmiques ni grans conseqüències si la eina no és econòmicament exitosa. Això em permet fer eines que estudis grans no tenen temps, ja que mentre estudis més grans tenen un temps limitat per fer les eines, aleshores desenvoluparan eines que els hi serveixin a ells i per al seu projecte, però aquest projecte al tractar-se de un treball de fi de grau, em puc permetre el luxe de desenvolupar una eina que englobi a més projectes amb diferents necessitats.

3.4. Riscos i pla de contingències

Tenint en compte tots els riscos mencionats anteriorment en el DAFO, s'han plantejat els següents plans de contingència:

- **Manca de coneixement:** La manca de coneixement anterior i experiència sobre aquest tema pot fer que diferents tasques mencionades en la gestió del projecte durin més del que està presentat en el apartat 3.1, la qual cosa pot causar grans canvis en el desenvolupament de l'eina. És per això que s'han presentat tasques a final de projecte com "Tool improvements and extra features" o "Code an Tool Clean Up", que encara que són importants, no són de vital importància, i es pot utilitzar el temps plantejat d'aquestes en finalitzar d'altres pendents que no permetin avançar el projecte.
- **Poc intuïtiu:** Aquest risc es pot solucionar fent test a gent externa per tal de veure si la interfície dissenyada és poc intuïtiva o funcional.

3.5. Anàlisis inicial de costos

En la següent taula es poden observar el anàlisis inicial de costos que s'ha calculat per al projecte. Per a aquest anàlisi s'han tingut en compte els següents aspectes:

- El desenvolupament de la eina complerta dura 6 mesos.
- El desenvolupador treballa en una oficina i cobra 900€ al més.
- El artista és comissionat per fer el tileset temporal i el final per 200€.
- El software usat pot variar depenent de les decisions que es prenguin a la hora de fer la tool.
- S'ignoren els costos de màrqueting de la eina.

		Cost/Salaries	Type	Amortization	Total cost for 6 months
Personal	Programmer	900€	Monthly	-	5.400€
	Artist	200€	Unique	-	200€
Equipment	Computer	900€	Amortization	3	150€
	Screen	150€	Amortization	3	25€
	Keyboard	30€	Amortization	3	5€
	Mouse	20€	Amortization	3	3€
	Table	100€	Amortization	6	8€
	Chair	70€	Amortization	6	6€
Software	Houdini	269€	Yearly	-	269€
	Unity	25€	Monthly	-	150€
	3ds Max	248€	Monthly	-	1.488€
	Substance Painter	20€	Monthly	-	120€
Manteinance	Taxes	50€	Monthly	-	300€
	Rent	600€	Monthly	-	3.600€
	Food	200€	Monthly	-	1.200€
				Total Costs	**12.925€**

Taula 2. Taula de gastos i costos del projecte

Així doncs, es pot veure com gran percentatge dels costos va cap al desenvolupador i cap a les llicències de programa. Si es volguessin reduir costos es podria usar Blender (que és gratuït) en comptes de 3ds max a la hora de modelar i es reduiria 1.488€ en costos.

4. Metodologia

La metodologia de treball que s'usarà per aquest projecte és la metodologia àgil de Scrum. Aquesta metodologia es basa en esprints, i ja tenim experiència prèvia amb projectes realitzats a la universitat. En aquest cas, en comptes de basar els esprints cada setmana o cada un període de temps fixe, el que es farà es basar els esprints en el temps plantejat per implementar una feature. És a dir, les features en las quals he dividit el diagrama de Gantt seran els esprints. Més concretament em basaré en els esprints del grup de "Tool Development". Per cada esprint es seguiran els següents passos:

- Primer de tot es plantejarà la meta o milestone en la que es vol arribar en aquest esprint.
- Seguidament es dividirà aquesta meta en una llista de tasques o característiques que necessitarem desenvolupar per arribar-hi.
- Finalment s'organitzaran les tasques depenent de la seva importància per al resultat final i el seu temps de desenvolupament. Amb això aconseguirem una llista de tasques amb diferents prioritats que podrem seguir en aquest esprint.

Si un esprint no es aconsegueix acabar a temps i ha quedat alguna feature crucial per al desenvolupament del següent, es tirarà enrere aquesta feature al següent esprint es faran ajustaments de temps en els següents perquè el projecte compleixi les dates límits determinades.

Per tal de assignar aquestes tasques dins de cada esprint, s'utilitzarà la eina de HacknPlan, molt similar a Trello, amb la qual ja tenim experiència de anteriors projectes i ens permet crear tasques i organitzar-les en columnes de "To do", "In progress" i "Done" de una manera molt visual i esquemàtica.

5. Desenvolupament del projecte

5.1. Anàlisis previ

Per tal de començar a desenvolupar una eina com la que es vol desenvolupar, primer s'ha de entendre bé com funcionen els sistemes modulars i quina serà la nostra aproximació.

Com bé ens explica el seu nom, els sistemes modulars es tracten de sistemes basats en peces o mòduls. Generalment son varis models que encaixen entre si, i ajuntant-los pots generar edificis o entorns.

Figura 5: Pack modular "Fantasy Interior Pack" del Unreal Marketplace

A la hora de fer la meva aproximació per la eina, els he separat et dos grans grups. Els que he anomenat: **Core parts**, i **Details** o **Accessories.**

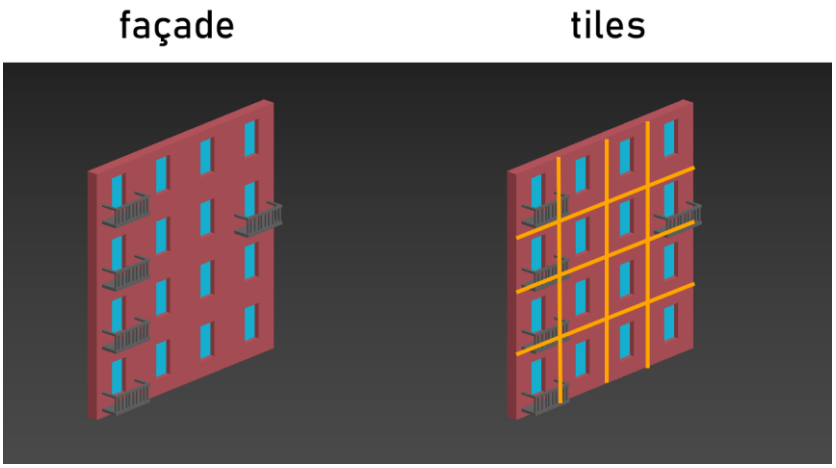


Figura 6: Exemple de una façana construida amb tiles.[1]

[1] Tant la figura 6 com la figura 7 han estat fetes per mi amb el software 3ds Max.

Les parts Core són les que es fan servir per fer la part base del edifici, és a dir, parets, sostres, terres, finestres, portes...

I les parts Detail són les que es fan servir com a complement de les anteriors, però que no són completament necessaris per a fer la estructura base, com estanteries, balcons, tendals...

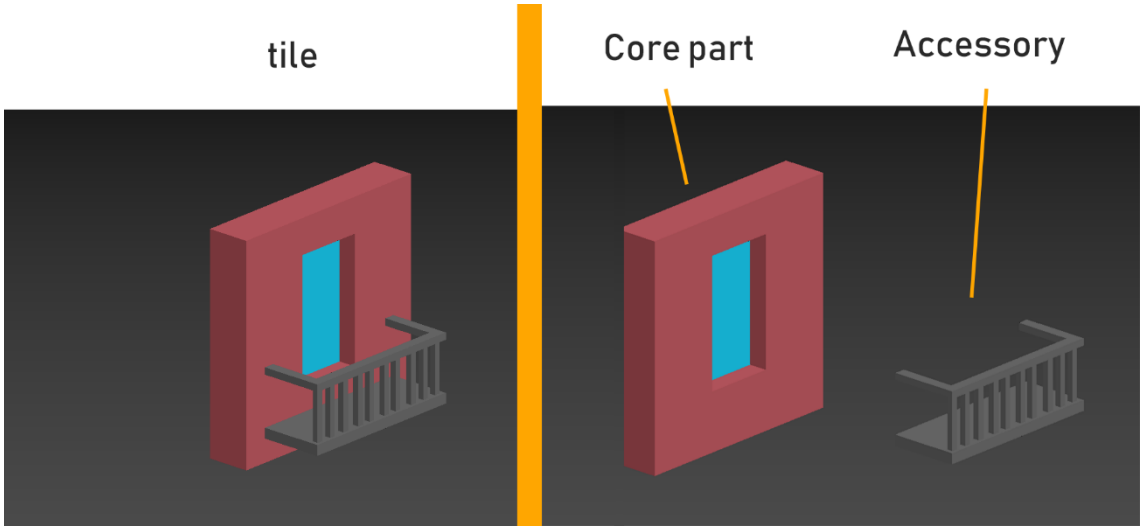


Figura 7: Exemple dels 2 grans grups en els que he dividit les tiles

5.2. Software a utilitzar

Per a desenvolupar aquesta eina s'han analitzat les següents opcions:

- **Houdini Engine:** Com s'ha mencionat en apartats anteriors, encara que el workflow de Houdini sigui diferent al clàssic, si es plantegen correctament, es poden desenvolupar eines que encaixin dins del workflow clàssic. Com a avantatges podem trobar que la eina que es desenvolupi servirà tant per a Unreal Engine com per Unity, ja que es pot importar als 2 motors, i que es pot veure visualment el progrés que vas fent, la qual cosa et permet veure amb més claredat com s'ha de programar certes coses.
  Com a principal desavantatge és el fet que es necessari com a mínim una llicència indie de Houdini per fer servir la eina dins dels motors.
- **Unity:** Crear una eina per a Unity és una altra opció. Unity et permet programar noves interfícies, la qual cosa es un gran avantatge a la hora de desenvolupar una eina mínimament visual. Per la contra, com a desavantatge trobem que no té cap mena de feedback visual a la hora de programar, i programar una eina

d'aquest volum, on el usuari pot canviar moltes coses pot ser molt complex sense veure pas a pas el que estàs realitzant. A part d'això, la eina sol funcionaria amb Unity.

- **Unreal Engine 4:** L'altre motor de videojocs per excel·lència és Unreal Engine 4. Aquest motor, a part de poder programar de la manera clàssica, té la opció de programar visualment, la qual cosa fa molt més fàcil controlar les diferents variables que podria tenir una eina com aquesta mentre es desenvolupa. Com a desavantatge però, tenim el mateix problema que amb Unity. A part de que la eina sol seria compatible amb aquest motor, Unreal al ser un motor de videojocs, està pensat per fer videojocs, i no per generar geometria en el espai, amb la qual cosa encara que sigui més visual que Unity, tampoc pots veure pas a pas el que s'està realitzant.

A partir de aquestes tres opcions s'ha decidit fer servir **Houdini** per tal de desenvolupar la eina. Les principals causes són:

- **Exportable a Unity i a Unreal:** gràcies a Houdini Engine, es pot exportar aquesta eina a ambdós motors, amb la qual cosa satisfà el fet de que la eina sigui el més generalista possible.
- **Fàcil de testejar:** gràcies a que Houdini et permet veure que es fa a cada pas visualment, ens permetrà anar molt més ràpid a la hora de desenvolupar una eina, la qual cosa alleugera molt el temps de desenvolupament.

5.3. Desenvolupament de la eina

A partir d'aquí començo a desenvolupar la base en la que es basarà la eina.

El software que faré servir serà Houdini, ja que això em permetrà fer una eina que serà fàcilment importable a Unity i Unreal, fent la així compatible amb tilesets i packs ja existents i em permetrà agilitzar el procés del desenvolupament.

Per començar a desenvolupar la base, començaré desenvolupant edificis amb la forma més simple i més quotidiana que ens podem trobar a la vida real, un cub. Primer de tot generaré un cub de punts en el espai. Segons en la cara que es trobin aquests punts generarem les seves normals, i aquests punt, junt amb les seves normals dictaran a on i en quina direcció es posicionaran els mòduls. Per tal de que la eina sigui el més flexible possible, se li dona la oportunitat al usuari decidir la altura, llargada i amplada d'aquest cub, junt amb la distància vertical i horitzontal que podem trobar entre punts.

D'aquesta manera s'aconsegueix que la eina serveixi per el màxim de tilesets o packs possibles.

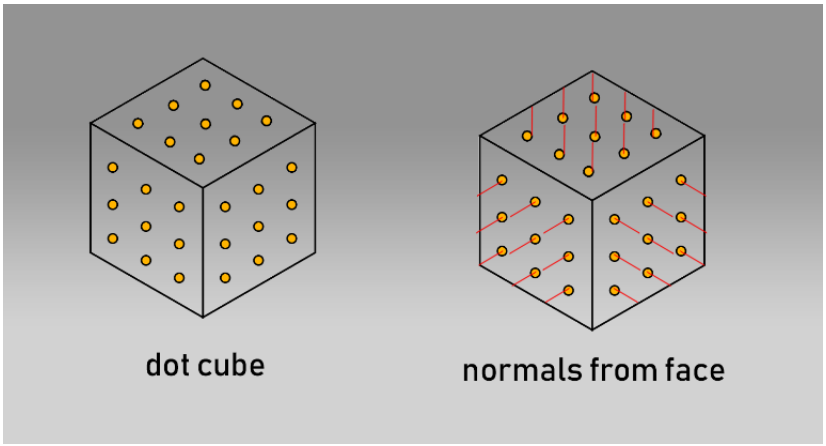


Figura 8: Point Cube i Point Cube amb normals.

Un cop tenim aquest cub en el qual podem posicionar les diferents peces, fa falta veure com les posicionarem, és a dir, com posicionarem les diferents variacions com parets amb finestra, sense finestra, porta, etc.. i quines variables podem posar per tal de aconseguir la millor similitud possible amb un edifici. Això es farà a partir de generar diferents grups de punts dins de aquest cub i omplint cada grup amb una tile diferent. La qüestió es analitzar quins grups s'han de generar i com.

Per tal d'aconseguir això, el procés que s'ha fet és analitzar edificis de la vida real i intentar desglossar-los en tiles.

Figura 9: Edifici dividit en tiles

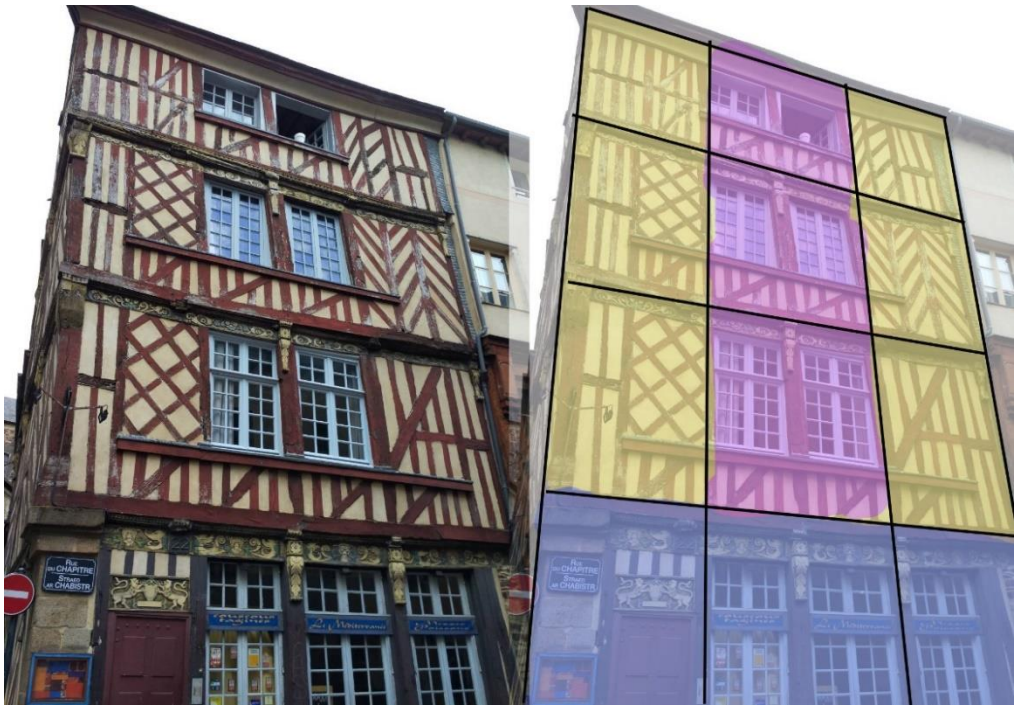


Figura 10: Altre exemple de edifici dividit en tiles

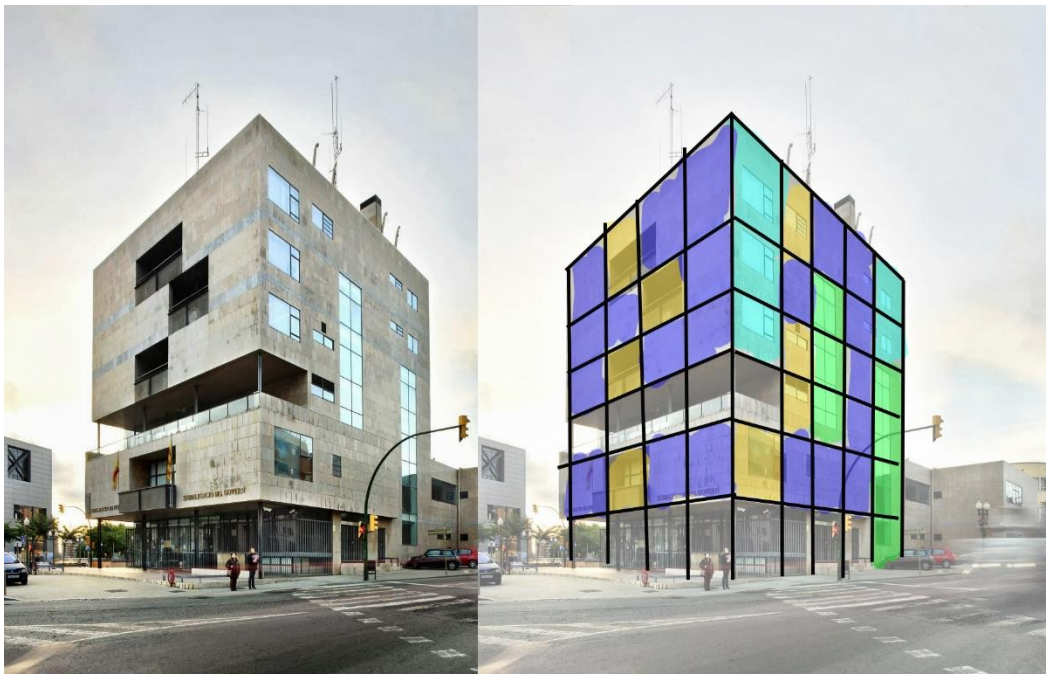


Figura 11: Altre exemple de edifici dividit en tiles

Un cop analitzades moltes imatges de edificis reals es pot arribar a una conclusió. Si analitzem els exemples exposats anteriorment podem veure com els edificis es poden desglossar amb tiles, i els colors de les imatges representen els diferents grups en els quals es poden desglossar (finestres, parets de primera planta, parets de segona planta...)

Si comparem les imatges amb motles d'altres extraurem que en edificis es poden treure varis patrons similars. Apareixen els següents patrons:

- **Patrons de pisos:** Es pot veure com hi ha patrons horitzontals que sol segueixen un sol pis.
- **Patrons de columna:** Hi ha patrons de finestres o parets que es repeteixen verticalment.
- **Patrons aleatoris:** Hi ha edificis que no segueixen cap patró en específic, i tenen tiles repartides mes o menys aleatòries. Sobretot es sol trobar en façanes interiors o cases velles.
- **Patrons de façana:** Potser el més important i el que més s'obvia. Molts edificis al tenir un altre al costat, moltes vegades hi ha patrons (com finestres) que sol apareixen en la façana principal.
- **Sub-patrons aleatoris:** Dins dels diferents grups de tiles, podem trobar patrons aleatoris, com per exemple de un grup de finestres, que certes d'aquestes estiguin amb persianes baixades.

Així doncs, serà necessari recrear aquests patrons dins de Houdini per tal de que el usuari pugui escollir quins fer servir a la hora de desenvolupar els seus edificis.

El sistema que em plantejo doncs, es un sistema on el usuari pugi escollir el numero de grup de Core tiles que vol fer servir per el edifici, i que pugui manipular cada un d'ells al seu gust. Així que abans de desenvolupar varis grups a la vegada, es començarà desenvolupant 2, i un cop aquests estiguin desenvolupats, es podran duplicar tantes vegades com el usuari vulgui.

Per començar a desenvolupar-lo he escollit els 2 grups més basics a la hora de fer un edifici: parets amb finestra, i parets sense finestra (tot i que, aquests dos grups poden ser el que el usuari desitgi). A partir d'aquí s'ha començat a desenvolupar la eina.

En les següents imatges que veureu es representen aquests dos grups amb models placeholder que s'han fet servir per treballar amb la eina. Un cop importada l'eina dins de un motor de videojocs, aquests models es poden canviar per els que el usuari desitgi.

5.3.1. Emmascarament de façana

Primer de tot començarem amb els grups de façana. Per tal de aconseguir això, el que es fa es emmascarar i agrupar els grups que ja tenim segons les seves normals. El usuari podrà escollir quin eix és el que vol emmascarar, i com que els punts de cada façana apunten a una direcció diferent, si s'emmascara els punts que tinguin una normal mirant cap aquella direcció, i per tant aconseguirem emmascarar una façana.

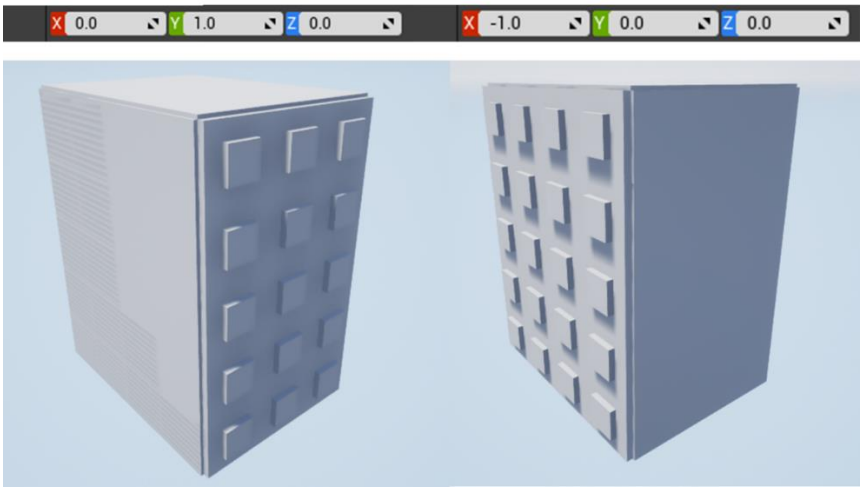


Figura 12: Prototip de masking de façanes funcionant a Unreal Engine

A part d'això, també s'ha tingut en compte de que si el usuari vol emmascarar més d'una façana (posem el cas que el edifici faci cantonada), pot fer-ho posant les direccions de les dues façanes, i augmentant la variable *Spread Angle* a 45º, la qual cosa permetrà agafar les normals de ambdues façanes.

Figura 13: Exemple de enmascarament de dues façanes

5.3.2. Emmascarament horitzontal i vertical

Per altra banda, per tal de emmascarar punts de un sol pis o columna, s'ha seguit un procediment diferent. El que s'ha fet per tal de emmascarar aquest punts es crear una caixa de la mida de un pis, la qual s'adapta a la amplada i llargada del edifici, i emmascarar els punts que colisionin amb aquesta. La posició de aquesta caixa depèn del input del usuari, el qual pot decidir quin pis emmascarar.

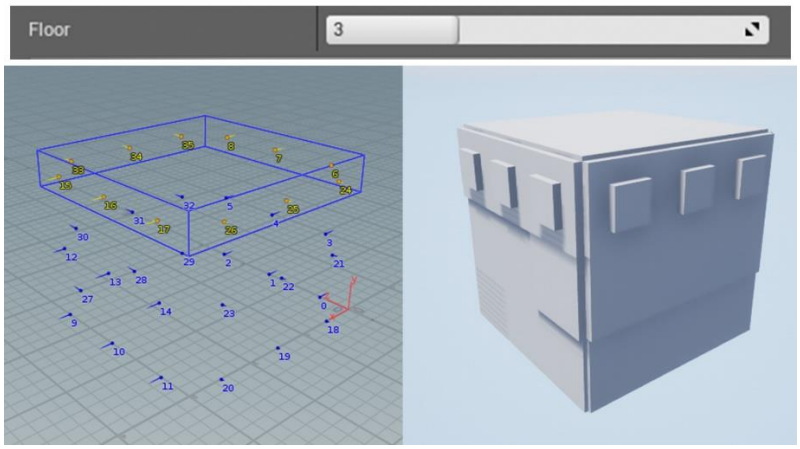


Figura 14: Exemple funcional de emmascarament per pisos

5.3.3. Emmascarament aleatori

Finalment ens trobem amb l'emmascarament aleatori. Per tal de aconseguir fer aquest emmascarament, s'aplica una funció que escull punts aleatoris i en crea un nou grup. Com que un grup aleatori no es pot controlar, i es vol que la eina sigui el més útil possible, s'aplica una seed a partir de la qual s'agrupen els punts, de tal manera que si el usuari no li convenç el grup creat, pot canviar la seed, i obtindrà resultats diferents.

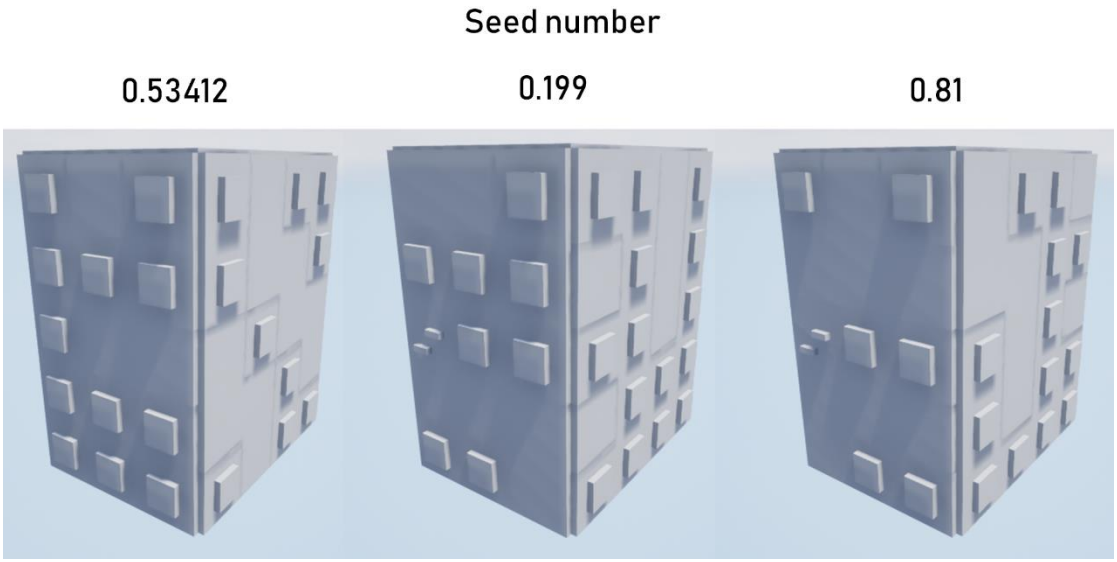


Figura 15: 3 grups aleatoris creats amb diferents seeds

A part d'això, també pot ser interesant controlar mínimament la quantitat de punts que s'emmascaren, ja que es probable que el usuari vulgui reduir el nombre de peçes que apareixen, o per el contrari incrementar-les. Així doncs, s'ha posat al abast de l'usuari una variable que té un rang de 0 a 1 que li permet decidir aproximadament quants punts s'emmascararan, sent 0 cap punt, i 1 tots els punts.

Figura 16: 3 exemples amb diferents quantitats de finestres

Finalment, per cada grup restant, s'ha tornat a afegir una mascara random la qual permetrà posar varies tiles en un grup. Posem per exemple que hem creat un grup de finestres, i ens interessa que algunes d'aquestes tinguin persianes baixades, o que una paret tingui variacions de maons. Gràcies a aquesta ultima mascara random, ens permetrà passar aquestes variacions de tiles i es posaran de manera aleatòria. Igual que la mascara aleatòria que s'ha explicat anteriorment, aquesta també permetrà el usuari decidir la quantitat de variacions que s'aplica a la mascara.

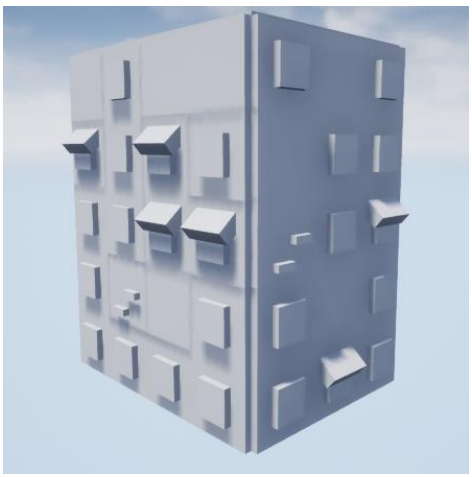


Figura 17: Un exemple de un grup de finestres amb variacions de tendals, i un grup de parets amb variacions de parets amb maons

Un cop tenim aquests quatre grups, fer que tots aquests funcionin a la vegada i que es puguin activar o desactivar cada un per separat és on es genera una gran complexitat a la hora de desenvolupar la eina. Al final ens hem de quedar amb un sol objecte que ens permeti anar alterant entre grups segons l'input de l'usuari.

Finalment s'ha aconseguit que cada un dels grups complementi als altres, i que si algun d'ells està desactivat, s'ignori. Així doncs si per exemple s'activa el agrupament de façana i el aleatori, sol es farà en la façana desitjada, i així conseqüentment amb la resta de grups.

En el cas especial de que s'activin un grup de columna i un horitzontal, en comptes de restar-se un al altre, es complementaran, formant així ambdues formes.

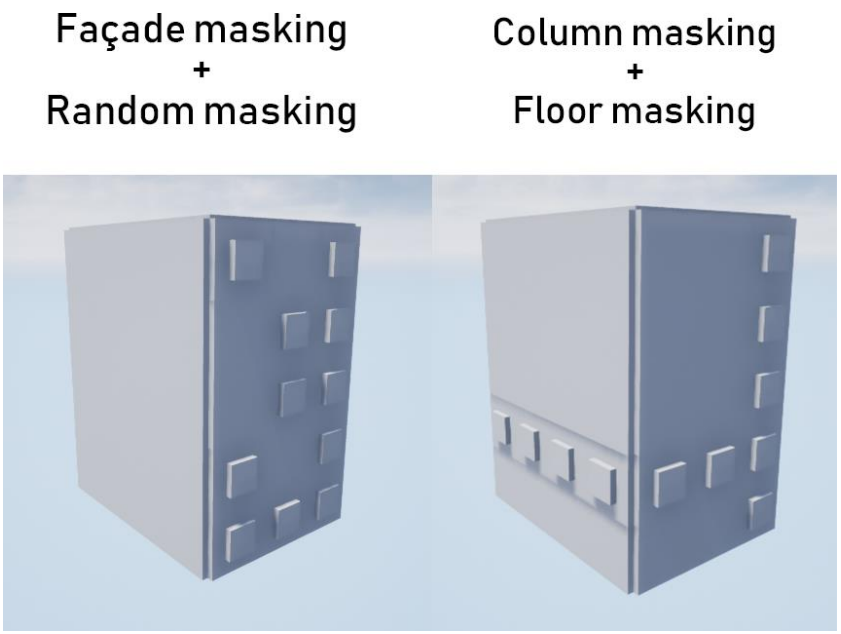


Figura 18: Exemples varis de fusió de màscares.

Així docs, ja tenim la base de la eina creada. De moment tenim creat el sistema per fer grups de les parts Core de les tiles. Crear aquest primer grup és potser la part mes complexa de la eina i la que porta més temps de desenvolupament, però un cop fet això ja tenim una gran i major part de la eina ja creada. Sol amb els grups mostrats ja tenim una gran quantitat de variables per fer molts edificis.

La gràcia d'aquest grup, és que un cop creat un, es poden crear tants com el usuari vulgui, ja que el procediment és el mateix. En el exemple que s'ha fet servir mentre es desenvolupava, s'ha explicat per fer parets i finestres, però si es vol, el usuari podria crear un tercer grup on s'emmascarés aleatòriament 1 tile del primer pis, i li serviria també per crear portes. D'aquesta manera, li donem molt poder al usuari per crear els seus edificis de una manera procedural i ràpida.

De la mateixa manera, es pot extreure aquest grup a la hora de posar els Details al edifici. Per exemple si es vol posar que certes finestres(Core) tinguin baranes(Detail), es pot crear un grup aleatori a partir del grup de les finestres el qual posarà baranes, i gràcies a com esta fet el emmascarament, el usuari podrà decidir la quantitat finestres tindran barana gracies als paràmetres de "Random Seed" i "Random Quantity" explicats anteriorment (Emmascarament aleatori).

5.3.4. Punts de millora

Ara de moment tenim una eina que ens permet desenvolupar blocs de pisos amb moltes variacions. Els següents passos a seguir per acabar la eina bàsica són els que s'han mencionat anteriorment: crear **múltiples grups**, i crear **grups per Details**.

Un cop s'hagin desenvolupat aquests punts la eina bàsica ja estarà acabada. Tot i així és important tenir en compte els següents punts de millora que que tenen la possibilitat d'implementar-se si la planificació ens ho permet. S'ha de tenir en compte que cada un d'aquests punts té una complexitat molt alta, i cada un d'ells suposaria un temps de desenvolupament extra.

Els punts de millora que em plantejo són els següents:

- **Forma base no cúbica:** De moment podem crear blocs de pisos i cases amb la limitació que la forma que generarem serà cubica. Seria interesant aconseguir crear formes mes o menys aleatòries ja que ens permetria crear edificis amb molta més varietat de forma.
- **Interiors:** Crear interiors comporta una alta complexitat a la hora de crear els edificis, però també pot ser una característica molt útil.
- **Sostres no plans:** Crear sostres no plans també representa una gran complexitat, però també és una característica que seria molt util a la hora de donar més variació a la eina.

5.3.5. Neteja de codi i implementació de tots els grups.

Abans de decidir quina millora es vol implementar el que s'ha fet ha sigut una neteja de codi i la implementació final dels diferents grups. Mentre que anteriorment s'ha explicat les diverses opcions de emmascarament i aleatorització que té un grup, ara a la hora de desenvolupar la eina s'ha de duplicar aquest grup uns 5 cops per donar-li al usuari prou varietat perquè pugui generar els edificis que vulgui. En un principi, s'havia determinat poc temps en realitzar aquesta tasca ja que tota la lògica i la part difícil de generar un grup ja estava feta. La realitat ha sigut que tot i no ser una tasca difícil, ha sigut molt llarga i ha ocupat bastant temps, ja que s'ha passat de tenir que enllaçar internament 30 variables que conté un grup, a més de 150 variables que contenen els 5 grups. Això ha fet que el temps dedicat a la tasca hagi sigut molt major.

Tot i que en aquest document no s'ha explicat amb detall quins nodes i funcions específiques s'han utilitzat dins del software Houdini (ja que aquest treball tracta del desenvolupament de la eina i de la teoria desenvolupada darrere d'ella, i no sobre un tutorial de com utilitzar Houdini), en la següent figura podem veure la diferència de complexitat del graf de la eina amb un grup base (esquerra), i el graf de la eina amb 5 grups diferents (dreta).

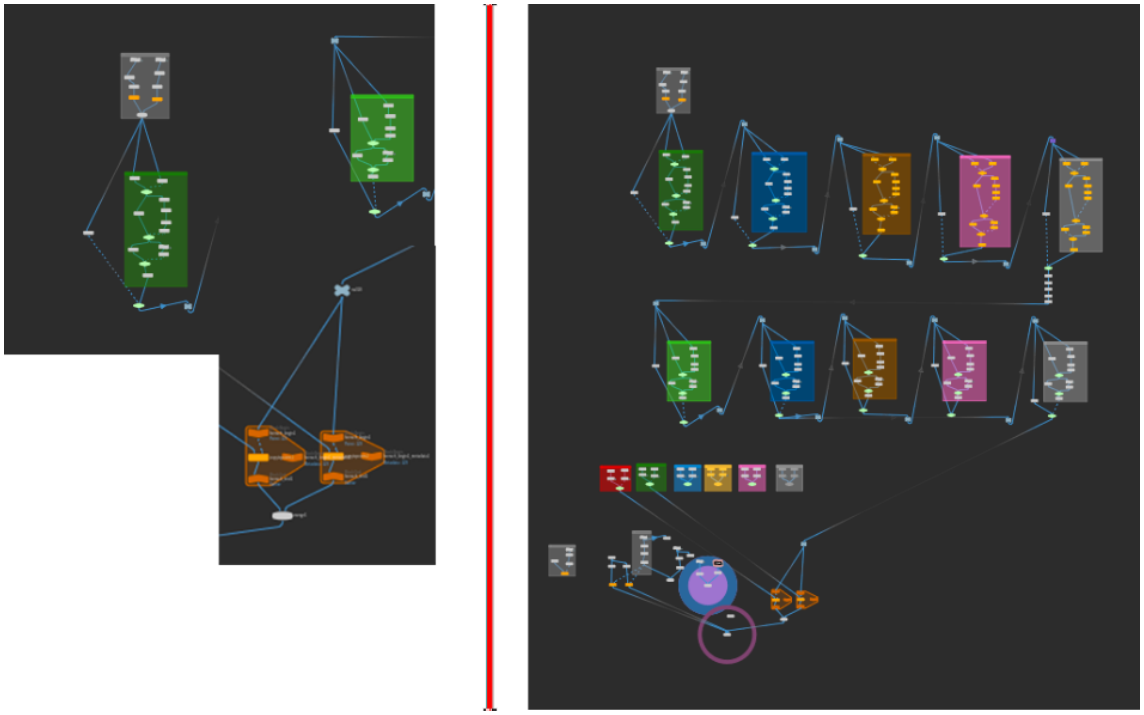


Figura 19: Dreta: Graf del projecte amb un sol grup (esquerra) Graf del projecte amb 5 grups de base i detall

5.3.6. Elecció dels punts de millora

Un cop s'ha tingut la base de la eina feta i polida, analitzant el temps restant que es té plantejat de desenvolupament i les 3 opcions de punts de millora s'ha escollit quin dels 3 fer. La elecció ha sigut la realització de **sostres amb pendent/ no plans**.

S'ha escollit aquesta opció tenint en compte els següents factors:

- La creació de interiors suposa replantejar la base de la eina, i tenint en compte el temps restant, hi ha una gran possibilitat que no s'assoleixi un mínim base.
- La creació de formes no cúbica té fàcil solució, ja que la eina es pot fer servir per crear diferents parts com diferents edificis i després ajuntar-los a mà.
- La creació de sostres amb pendent/no plans no té solució alternativa, i permet que la eina sigui molt més generalista, podent crear cases amb estructures diferents.

5.3.7. Creació de sostres amb pendent

Crear un sostre amb pendent no es tan senzill com sembla i comporta moltes dificultats que a simple vista es solen ignorar. Els passos a seguir per generar aquests sostres han sigut els següents.

Primer de tot, si s'afegeixen sostres triangulars significa que l'alçada del edifici augmentarà, per tant s'hauran de posar més parets al costats que s'hauran d'agrupar amb els altres grups de parets (color vermell).

El següent pas és posar parets triangulars perquè encaixin amb els sostres amb pendent, i també un altre tipo extra de paret per la fila de sostre central (color groc).

Finalment s'afegeixen els sostres amb pendent (color blau) i els sostres amb canvi de pendent (grocs).

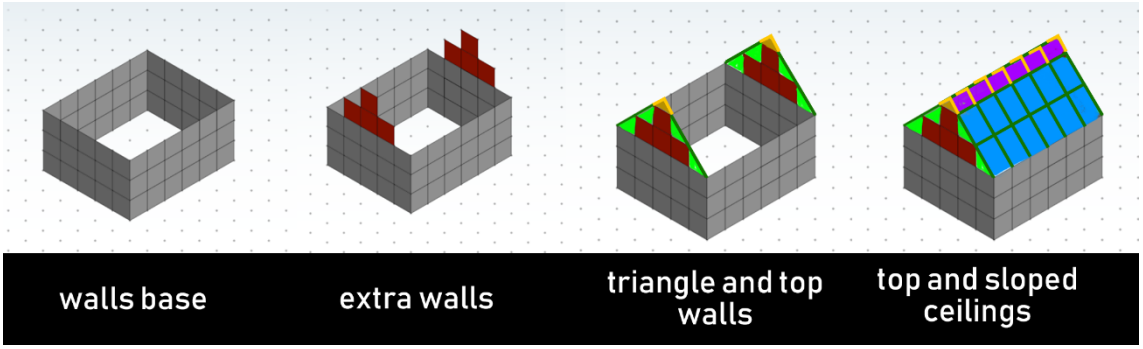


Figura 20: Passos a seguir en el desenvolupament de la millora de sostres amb pendent.

A part del fet que estem afegint 4 noves peces que es posicionen de manera diferent a la resta ja plantejades, ens trobem amb un altre dificultat. Com que el usuari decideix la llargada de la casa, per conseqüència això afecta al sostre de la casa, i si la llargada de la casa no és un numero senar (1,3,5,7), ens trobarem que el triangle generat no es perfecte i tots els grups fets per fer un sostre amb canvi de pendent (grup groc i lila) no apareixen, com es mostra a la següent imatge:

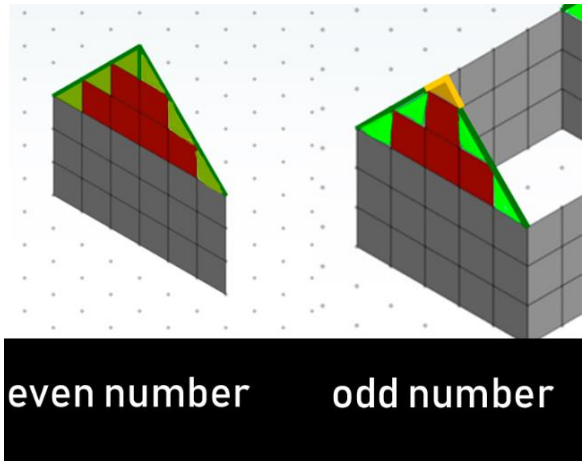


Figura 21: Diferència de estructura entre nombres pars o senars de llargada.

Això significa que depenent de el input que generi el usuari, els punts es posicionaran d'una manera diferent, i això fa que la programació de la posició d'aquests punts sigui molt més difícil. Tot i així, gràcies a la rapida iteració que et permet fer Houdini, s'ha aconseguit un codi òptim que adapta el sostre al input posat per el usuari.

5.3.8. Generació de assets modulars.

A la vegada que s'ha començat a desenvolupar la millora dels sostres amb pendent, s'ha començat a desenvolupar un pack d'assets modulars que em permetin ensenyar la utilitat d'aquesta dins de un motor de videojocs. He decidit fer un pack d'assets per realitzar una casa medieval ja que amb aquestes cases es poden mostrar sostres amb pendent, i moltes vegades segueixen patrons molt més complicats que edificis més moderns. Per tal de desenvolupar aquests assets, he utilitzat 3ds Max com a eina per a modelar i Substance Painter com a eina de texturitzat. El pack consta de 9 variacions de paret amb 3 variacions de finestres i 2 variacions de detalls. El pack també consta amb 2 variacions de sostres i parets que s'adapten a aquests.

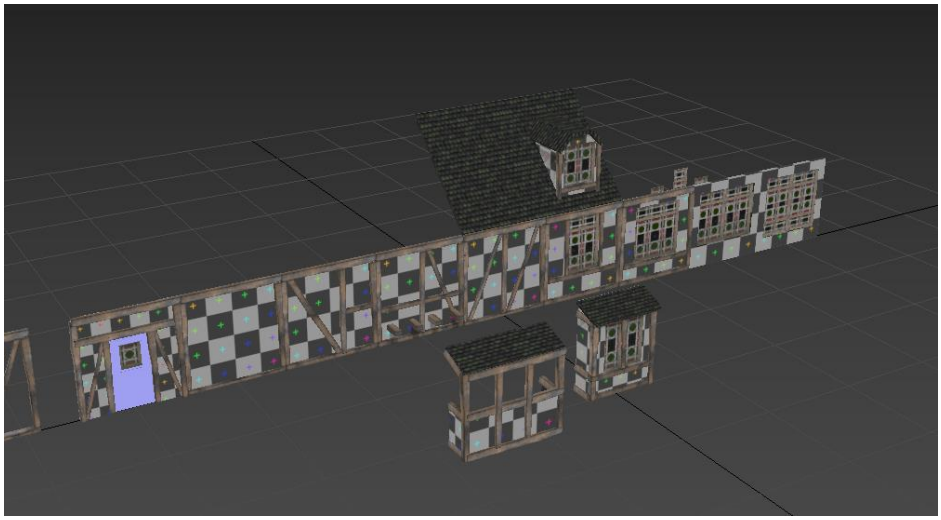


Figura 22: Imatge preliminar del conjunt de assets amb textures temporals.

Figura 23: Imatge de exemple del texturitzat que s'ha utilitzat per aquest pack.

5.3.9. Problemes de instàncies

Un cop acabat el desenvolupament de la eina, s'han fet uns testeig intern abans de fer els externs per tal de comprovar que la eina funcionés correctament. Tot i que la eina es va testejar prèviament quan estava menys desenvolupada, no es va trobar amb un problema el qual s'ha trobat testejant-la en profunditat. La eina ha complert el objectiu, ja que redueix el temps que es tarda en construir una casa si ho comparem amb el mètode tradicional, però apareix un gran problema. Si per exemple volem crear una segona casa a partir de una ja creada i canviem certs paràmetres (com afegir més grups, canviar el sostre de pla a amb pendent.,etc) ens trobarem que els mòduls instanciats es canviaran de posició (per exemple, hi haurà parets que es posaran el la posició de sostres, etc..).

El problema que se s'ha trobat és causat per com Houdini Engine instància els models als punts designats. Si es varia el nombre de objectes instanciats, encara que s'hagin designat anteriorment els models a aquells punts específicament, Houdini Engine no actualitzarà els grups fent que aquests es canviïn de orde.

Posem per exemple que en un primer cas estem instanciant 4 objectes (sostre, paret 1, paret 2 i paret 3) i volem fer que aquesta casa tingui un sostre amb pendent en el cas 2, la qual cosa ens comporta instanciar 5 objectes (sostre, paret especial, paret 1, paret 2, paret 3). En el primer cas, si ja s'han canviat els models, la casa es tindria que veure a la perfecció ja que els models s'estan instanciant en els grups que toquen. En canvi a la casa 2, quan s'afegeix un cinquè grup, en comptes de actualitzar les instàncies, te les deixa en el ordre del primer. Això causa que tots els models s'hagin canviat de grup i es tinguin que canviar manualment per tal de reorganitzar-los i que la casa torni a quedar bé. Això passa doncs, cada cop que es vulgui instanciar un nou grup de models(sigui canviar sostre, afegir detalls o nous grups).

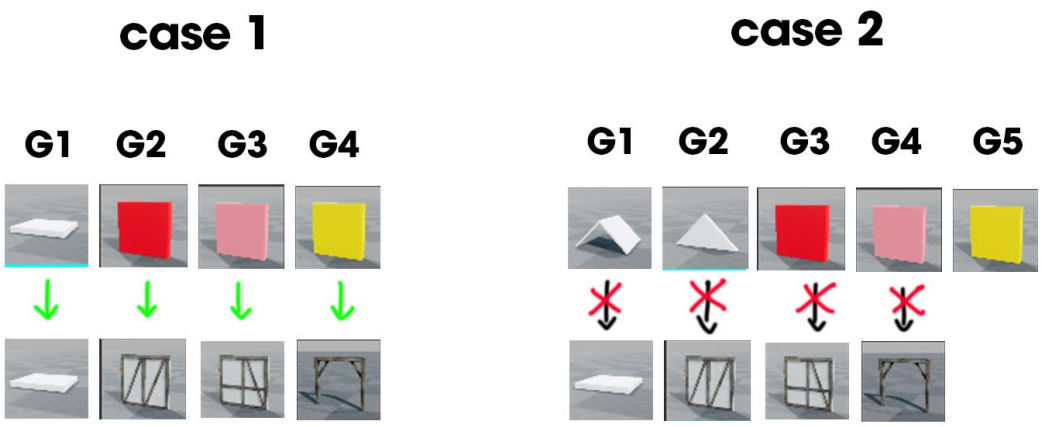


Figura 24: Casos mencionats del problema amb les instàncies.

5.3.10. Forçar instàncies.

Tot i que la eina seguia sent més eficient que el mètode tradicional (tardant aproximadament 3-4 minuts en reorganitzar els models comparats amb els 6 que es tarda en posar els models extra manualment), la eina en sí perdia certa part de la seva utilitat, ja que la gràcia de desenvolupar una eina procedural és que s'adapti als inputs del usuari automàticament, i no que ho tinguis que fer tu manualment.
A partir d'aquí, apareix la opció de buscar una solució a aquest problema, fent un gran canvi dins a eina, o per altra banda, deixar-ho tal com està, ja que ja compleix amb els tests. S'ha escollit la primera opció, ja que encara que comporti molt temps solucionar-ho, si ens basem amb la planificació establerta i es té en compte el temps disponible per a desenvolupar, hi ha temps per desenvolupar-ho sense causar un gran retràs. A part d'això, si es pot solucionar tal problema, la eina serà molt més útil que la versió actual, ja que els temps que tarda l'usuari en crear cases serà dràsticament reduït.
És així doncs, com es decideix fer aquest canvi.
Per tal de solucionar aquest problema, la única solució possible que hi ha és forçar a Houdini Engine a instanciar els models que el usuari vol en el grup corresponent. Sense entrar en detalls tècnics molt específics, per tal d'aconseguir això, s'ha tingut que crear un atribut equivalent a cada grup. Aquest atribut conté una referencia[2] al model que es vol aplicar en aquell grup, i això fa que encara que es creïn nous grups, assegura que els models s'instanciïn en el seu grup. Amb això l'usuari ara sol té que referenciar els models que vol fer servir 1 cop, en comptes de canviar-ho cada vegada que es canvia un paràmetre de la casa.

5.3.11. Testeigs interns

Un cop s'ha tingut la eina desenvolupada, s'ha testejat per veure si complia amb els objectius proposats, sent el principal la reducció de temps a la hora de desenvolupar un entorn en un videojoc. Per tal de testejar això s'ha realitzat la següent prova. Construir dues cases de diferents complexitats amb la eina desenvolupada, i també seguint el procediment tradicional.

[2] S'entén com a referència, el directori del arxiu/model dins del projecte creat per l'usuari.

La primera casa que s'ha creat és la mostrada en la següent figura:

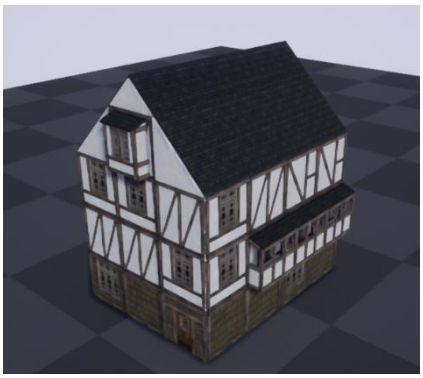


Figura 25: Primera casa realitzada en el testeig

Els resultats han sigut els següents:

- **Mètode amb la eina:** 8 minuts i 36 segons.
- **Mètode tradicional:** 11 minuts i 24 segons.

Com es pot comprovar, en aquest cas la metodologia amb la eina és quasi 3 minuts més eficient que el mètode tradicional. Tot i que la diferència de temps no és especialment elevada.

Com a segona prova, s'ha provat de crear, a partir de la primera casa, una casa 1 pis més alta i amb algunes variacions, com la figura mostrada:

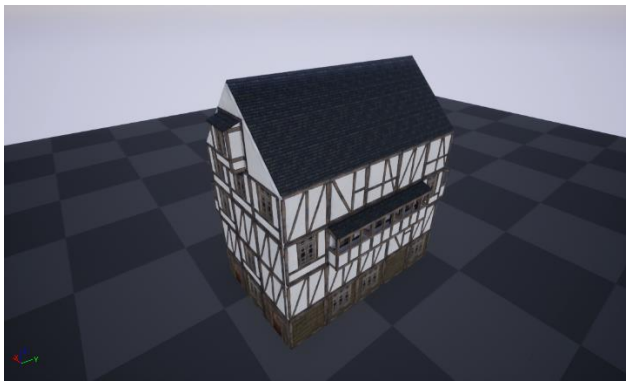


Figura 26: Segona casa desenvolupada en el testeig

Els resultats obtinguts han sigut els següents:

- **Mètode amb la eina:** 47 segons.
- **Mètode tradicional:** 4 minuts 16 segons.

Aquí es on es pot començar a veure el potencial de la eina per a desenvolupar escenaris. A la hora de fer petits canvis, ja sigui perquè s'ha de iterar la casa, o en vols crear una altra, com que sol s'han de tocar 2 paràmetres en comptes de anar col·locant nous models un a un, s'optimitza molt més el temps.

Finalment, per acabar el testeig s'ha desenvolupat la casa mostrada en la següent figura.

En aquest cas la casa consta de models completament diferents i té una diferent mida que les anteriors.

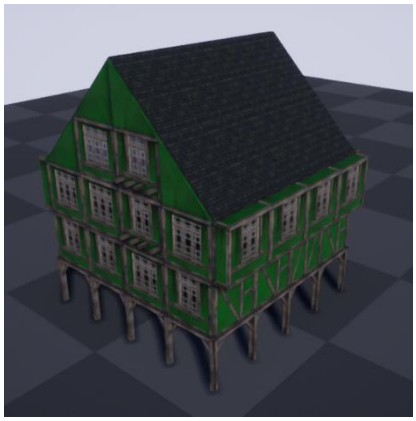


Figura 27: 3ra casa a realitzar en el test.

Els resultats obtinguts son els següents:

- **Mètode amb la eina:** 3 minuts 32 segons.
- **Mètode tradicional:** 12 minuts 16 segons.

En aquesta prova és on es mostra la gran utilitat de la eina. Gràcies a que amb la eina sol s'han tingut que canviar paràmetres i models, el temps que es tarda en construir una segona casa és molt menor a la que es tarda amb el mètode tradicional, ja que, amb aquest últim, s'ha de tornar a fer la casa des de 0.

5.3.12. Testeigs externs.

Per als testeigs externs s'ha realitzat la mateixa prova però demanant sol desenvolupar el primer i el tercer edifici mostrats anteriorment. Previ a la prova, se li ha fet al usuari un breu resum de com funciona la eina perquè entengui la base abans de testejar. S'ha estat present a la prova per tal de cronometrar el temps tardat.

Al acabar els usuaris han respòs un petit formulari que s'ha creat amb les següents preguntes:

- **Rol:** En quin rol es defineixen (artista, programador o dissenyador).
- **Temps 1:** Temps emprat per fer la primera casa amb la eina.
- **Temps 2:** Temps emprat per fer la primera casa amb la metodologia tradicional.
- **Temps 3:** Temps emprat per fer la segona casa amb la eina.
- **Temps 4:** Temps emprat per fer la segona casa amb la metodologia tradicional.
- **Hi ha alguna cosa que no s'hagi entès o que creguis que es pot millorar?:** Que creu que és millorable de la eina.

Els resultats que s'han tret d'aquests formularis són els següents:

En quin rol et defineixes?	Temps 1	Temps 2
Artista	00:10:12	00:12.24
Programador	00:09:47	00:13:29
Programador	00:10:37	00:11:56
DIssenyador	00:11:05	00:13:21

Taula 3: Taula de resultats part 1

Temps 3	Temps 4	• Hi ha alguna cosa que no s'hagi entès o que creguis que es pot millorar?
00:05:16	00:13:28	Creo que en la ui estaría bien ver mejor en que grupo te encuentras.
00:04:46	00:12:57	Estaría bien añadir grupos en el tejado
00:04:11	00:12:48	No.
00:06:17	00:14:42	Me gustaría tener más grupos.

Taula 4: Taula de Resultats part 2

Veient aquests resultats dels testeigs externs es poden treure les conclusions que, tot i que hi ha coses a millorar i que les diferències de temps són menors (comparats amb els testeigs interns), la eina segueix complint el principal objectiu que es proposava, que és agilitzar el procés de creació de entorns.

5.4. Estat actual del projecte

5.4.1. Rúbrica 2 (03-05-2019)

5.4.1.1. Seguiment de la metodologia i planificació

Respecte a la **metodologia** presentada al principi del projecte, s'ha seguit fent l'ús dels esprints i de la metodologia àgil de scrum, però hi ha hagut un petit canvi. Com s'havia mencionat, els esprints estaven basats en les tasques del Diagrama de Gantt. Això ja m'ha servit a mi per organitzar-me internament per tal d'arribar a aquesta meta. És per això que s'ha decidit no subdividir els esprints en tasques i posar-les al HacknPlan o Trello, ja que al final em suposava una pèrdua de temps crear les tasques i anar-les movent al llarg que es desenvolupaven. Així doncs, finalment s'ha descartat la eina de Trello i ens hem centrat més en els esprints posats en el Diagrama de Gantt.

Basats en la **planificació** establerta fins la entrega de la Rúbrica 2 en el punt **3.1.1**. no hi ha hagut cap canvi considerable en les tasques i esprints plantejats. La tasca de *testing* de la tool s'ha fet internament, ja que encara no està acabada i no és completament usable, però la resta de tasques s'han complert correctament tot i que la tasca de implementar algoritmes i variabilitat ha durat més del planificat, la qual cosa ha fet que el temps de testeig hagi sigut reduït de 6 a 3 dies (com ja est tenia contemplat en els plans de contingència).

Finalment doncs ja es té una demo de la eina completament funcional que ens permet crear edificis procedurals, com es tenia plantejat. Durant el temps que es té previst per l'esprint de "*Tool improvements and extra features*" per la entrega final, s'acabarà de desenvolupar la eina base i si hi ha temps s'intentarà implementar alguna de les millores comentades en el apartat **5.3.4**. i si això comporta més temps del previst, sempre es pot reduir el temps de "*Code and Tool Clean Up*" com a pla de contingència.

5.4.2. Entrega de seguiment (07-06-2019)

El estat actual del projecte es el següent. La base del projecte està completament implementada, i queda per realitzar la implementació completa de tots els grups de "Detail" i acabar la implementació de la millora del sostre amb pendent. Tot i que tant la tasca de implementar tots els grups base i la generació de sostres amb pendent ha durat més del planejat, la tasca de neteja de codi ha durat menys, ja que la base feta ja estava bastant polida de per si i no s'han tingut que fer grans canvis per optimitzar-la.
Així doncs podem concloure que a partir d'aquí no s'afegirà nou contingut a la eina i les següents dos setmanes que queden per la entrega final del projecte es dedicaran a acabar de implementar la millora de sostres amb pendent (ara es troba un 70% implementada), acabar de implementar els grups al sistema de "details". Finalment es testejarà la eina i es trauran les conclusions adients. També s'utilitzarà aquest temps per acabar de estructurar correctament el document de memòria del treball.

6. Conclusions i treballs futurs

Quan es va plantejar la idea del treball de fi de grau i em vaig decidir per aventurar-me en el mon procedural, tenia poc sinó inexistent coneixement sobre aquest tema. Gràcies a tota la recerca feta i el que s'ha après durant el període de desenvolupament del treball, crec que, tot i que és un tema molt extens i amb molt contingut, s'ha assolit unes bases sobre el contingut procedural que abans no es tenia.

Respecte a la eina desenvolupada, si s'analitzen els objectius tan específics com generals que s'havien plantejat al principi del treball, opino que, en menor o major mesura s'han assolit tots. Per una banda, personalment he après molt sobre aquest tema, i sobre quines complicacions comporta generar contingut procedural, i per altra banda si analitzem la eina, veient els resultats dels tests podem assolir que és una eina útil per a artistes i dissenyadors, els quals els hi permet crear contingut procedural sense que tinguin que tocar cap arxiu de codi.

Tot i que la eina desenvolupada es centra en la creació de edificis i que té certes limitacions, gràcies a la gran varietat d'opcions proporcionades al usuari, que s'han escollit després de una intensa i basta recerca, la eina permet crear tot tipus de cases, des de cases medievals fins a blocs de pisos de una ciutat.

Un cop acabat el treball de fi de grau i la primera versió de la eina i sabent quines són les seves limitacions, es poden plantejar varies millores que no s'han pogut desenvolupar per manca de temps o de coneixement. Els apartats de millora o expansió serien els següents:

- **Creació de interiors:** Sens dubte aquesta millora és la més complexa de les plantejades, ja que crear habitacions que tinguin sentit i que connectin correctament entre elles és una de les coses més difícils a reproduir. Tot i així, aquesta millora permetria a la eina obrir un nou ventall de possibilitats i per tant seria útil per a més jocs.
- **Més enllà de la creació de cases:** Si analitzem tot el procés de creació de la eina podem veure que es basa en instanciar (o copiar) models en punts en l'espai. La eina està centrada en la creació de edificis, però això no la limita a la creació d'aquests. Plantegem-nos, per exemple, que creem un edifici de 0 plantes. Això ens deixa amb un sostre pla i res més. Ara, si es decideix en comptes de posar models de sostre en aquests punts, posar models d'arbres. Si es poguessin agrupar bé, podríem generar un bosc de manera procedural.

Si es planteja d'aquesta manera, aquesta eina podria ser útil per crear moltes mes coses que edificis, i tècnicament no seria tan difícil.

- **Grups infinits:** Si s'hagués tingut més coneixement sobre el desenvolupament procedural, segurament s'hagués plantejat aquesta millora des de un principi. Ara mateix tenir un màxim de 5 grups és una limitació per a l'usuari, però per qüestió de temps, es van desenvolupar 6 (5 grups més el 0) perquè amb aquests, ja permetia en la majoria d'ocasions generar les combinacions desitjades. Ara bé, si es pogués fer de una manera que permetés a l'usuari crear un nombre de grups il·limitat, donaria a la eina molta més visibilitat.

Bibliografia / Webgrafia

- Alexandra, H. (18 / Setembre / 2016). *Kotaku*. Recollit de Kotaku: https://kotaku.com/a-look-at-how-no-mans-skys-procedural-generation-works-1787928446 Consultat: 20 Feb, 2019
- Carrier, E. (19 / Juny / 2018). *Youtube*. Recollit de Yotube: https://www.youtube.com/watch?v=NfizT369g60 Consultat: 21 Feb, 2019.
- Opara, A. (2016). *Vimeo* . Recollit de Vimeo: https://vimeo.com/188115734 Consultat: 21 Feb, 2019
- Wikipedia. (sense data). *Procedural Content Generation*. Recollit de Wikipedia: https://en.wikipedia.org/wiki/Procedural_generation Consultat: 21 Feb, 2019
- Contingut procedural dins de Unreal: https://www.youtube.com/watch?time_continue=557&v=J5LXIApQmzA Consultat: 8 Abr, 2019.
- Houdini Engine for Unreal https://www.sidefx.com/docs/unreal/_getting_started.html Consultat: 8 Abr, 2019.
- Houdini Forums https://www.sidefx.com/forum/topic/45275/ Consultat: 10 Abr, 2019.
- Houdini Engine for Unity & Unreal https://www.youtube.com/watch?v=a1tyOWugqUs Consultat: 14 Abr, 2019
- Houdini GDC https://www.youtube.com/watch?v=N4QMcpd66T0&t=735s Consultat: 14 Abr, 2019
- Houdini Engine Wiki: https://www.sidefx.com/docs/hengine2.0/_h_a_p_i__instancing.html Consultat: 24 Abr, 2019
- Houdini Global Variables: https://www.sidefx.com/docs/houdini/nodes/vop/global.html Consultat: 27 Abr, 2019
- Wiki de VEX https://www.sidefx.com/docs/houdini/vex/index.html Consultat: 27 Abr, 2019.
- Pregunta sobre VOPS als forums de Houdini https://forums.odforce.net/topic/11834-promote-parametervops/ Consultat: 27 Abr, 2019.

- Thread de Forums sobre agrupació aleatoria de punts https://forums.odforce.net/topic/14936-is-there-a-way-to-select-n-pointsprims-randomly-based-on-numberperce/ Consultat: 28 Abr. 2019.
- Documentació sobre Copy to Points http://www.sidefx.com/docs/houdini/nodes/sop/copytopoints Consultat: 28 Abr, 2019
- Documentació sobre com instanciar geometria a punts http://www.sidefx.com/docs/houdini/copy/instanceattrs.html Consultat: 28 Abr, 2019
- Documentació sobre el node Attribute Wrangle https://www.sidefx.com/docs/houdini/nodes/sop/attribwrangle.html Consultat: 28 Abr, 2019.
- Loops a Houdini https://www.sidefx.com/docs/houdini/shade/looping.html Consultat: 29 Abr, 2019.
- Tutorial sobre Houdini Digital Assets: http://tangdonna.blogspot.com/2011/04/digital-assets.html Consultat: 1 Maig, 2019.
- Tutorial de com copiar objectes aleatoris a punts en Houdini: https://www.youtube.com/watch?v=sO-mDdJBaVI Consultat: 1 Maig, 2019.
- Documentació de com referenciar paràmetres a Houdini: https://www.sidefx.com/docs/houdini/network/copying.html Consultat: 2 Maig, 2019.
- Tutorial de com instanciar objectes dins de loops: https://www.youtube.com/watch?v=mVim6iPEFGY Consultat: 14 Maig, 2019.
- Documentació sobre inputs en Houdini Engine: https://www.sidefx.com/docs/unreal/_inputs.html Consultat: 14 Maig, 2019.
- Creació i modelat procedural de edificis: https://80.lv/articles/006sdf-using-houdini-to-simulate-buildings/ Consultat: 2 Jun, 2019.
- Diferencies entre atributs VEX i variables: https://houdinitricks.com/vex-attributes-vs-variables/ Consultat: 2 Jun, 2019.
- Tutorial sobre implementació de Houdini en Unreal: https://www.youtube.com/watch?v=cdfNdbN47OA&t=1629s Consultat: 15 Jun, 2019.